(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
1 November 2007 (01.11.2007)

PCT

(10) International Publication Number
WO 2007/122495 A2

(51) International Patent Classification:
H04L 29/06 (2006.01) *G06F 9/48* (2006.01)
G06F 21/00 (2006.01)

(21) International Application Number:
PCT/IB2007/001052

(22) International Filing Date: 23 April 2007 (23.04.2007)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/793,934 21 April 2006 (21.04.2006) US

(71) Applicant *(for all designated States except US)*: AXALTO SA [FR/FR]; 6, rue de la Verrerie, F-92190 Meudon (FR).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: LU, HongQuian, Karen [US/FR]; c/o Axalto SA, Intellectual Property Dpt, 6 rue de la Verrerie, F-92190 Meudon (FR).

(74) Common Representative: AXALTO SA; c/o Lukasz WLODARCZYK, Intellectual Property DPT, 6, rue de la Verrerie, F-92190 Meudon (FR).

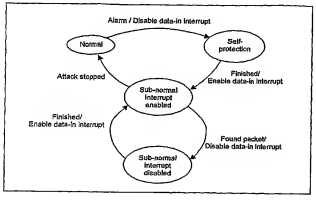
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, MT, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declaration under Rule 4.17:
— *of inventorship (Rule 4.17(iv))*

[Continued on next page]

(54) Title: A FRAMEWORK FOR PROTECTING RESOURCE-CONSTRAINED NETWORK DEVICES FROM DENIAL-OF-SERVICE ATTACKS



(57) Abstract: The invention relates to a resource constrained network device comprising detection means for detecting DOS attacks, and data-in interrupt means for notifying the device of network data input requests. The device comprises means to disable data-in interrupt means notifications when a DOS attack is detected.

WO 2007/122495 A2

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

A Framework for Protecting Resource-Constrained Network Devices from Denial-of-Service Attacks

The invention relates to a resource-constrained network device, and to a method for protecting such resource-constrained network device against DOS attacks (DOS stands for Denial of Service, a well known class of attacks).

Resource-constrained network devices are devices with embedded microprocessors, with very limited computing power and little memory resource, with networking capability, but having relatively low bandwidth. One example of such resource-constrained devices is the network smart card described in PCT/US2004/031572.

A smart card is a plastic card containing an integrated circuit with some memory and a microprocessor. The physical size of a smart card chip is limited to 25 mm2. The first generation of the Network Smart Card, demonstrated by Axalto, Inc. in year 2003, had 6K bytes of RAM and 512K bytes of flash memory. The speed of the microprocessor was 3.75 mHz. The smart card had a standard ISO 7816 interface, which is half-duplex, for communication. Recently, chip manufacturers have been adding full speed USB to smart card chips and have been making higher speed CPUs. For example, an upcoming smart card chip by ST Microelectronics has a full speed USB interface for communication, an expected 33 mHz CPU, 16K bytes of RAM and 64K bytes of EEPROM. Comparing to the state of the art computers, available resources for these small devices are still very limited.

These small devices join the Internet to provide services or to access resources. At the same time, they are exposed to network security threats just as other computers on the network. Because of the limited computing resources and lower network bandwidth, resource-constrained network devices are often more vulnerable to network attacks than other Internet nodes.

One of the network security threats is denial-of-service attack (DoS). Such attacks prevent legitimate users from accessing network resources, such as services on the Internet.

DoS attacks are described in details in particular in:

- Zwicky, E.D., Cooper, S. and Chapman D.B., Building Internet Firewalls, O'Reilly, 2000,
- Chang, R.K.C., "Defending Against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial," IEEE Communication Magazine, October 2002,
- "Denial of Service Attacks," CERT Coordination Center, http://www.cert.org/tech_tips/denial_of_service.html,
- Houle, K.J. and Weaver, G.M. "Trends in Denial of Service Attack Technology," CERT Coordination Center, October 2001, and
- "Distributed Denial of Service (DDoS) Attacks/tools," http://staff.washington.edu/dittrich/misc/ddos/.

The purpose of DoS attacks is to prevent or impair legitimate use of computer or network resources; for example, preventing users from access a popular Internet server. Common targets of resources are bandwidth, processing power, and storage capabilities. The most common DoS attack type is packet-flooding attack, which involves sending a large number of packets to a destination to cause excessive consumptions of resources.

In a distributed denial-of-service (DDoS) attack, a large number of compromised hosts are organized to send packets to a victim computer to consume excessively its resource and/or its Internet connection. There are two kinds of DDoS attacks: direct attacks and reflector attacks. In a direct attack, an attacker arranges many compromised hosts to send a large number of packets directly toward a victim. In a reflector attack, the attacker uses intermediary nodes (routers and servers), called reflectors, to launch the attack. The attacker arranges many compromised hosts to send packets that require responses to reflectors such that the packets' source addresses are set to the victim's IP address. Without realizing the plot, reflectors send responses to the victim consuming the victim's resource.

Common packet types used for DoS or DDoS flooding attacks include the following:

- TCP packets – A flood of TCP packets with various flags set are sent to the victim. Common flags include SYN, ACK, and RST.
- ICMP echo request/reply (also called Ping floods) – A flood of ICMP packets (echo request or echo reply) are sent to the victim.
- UDP packets – A flood of UDP packets are sent to the victim.

The DoS or DDoS attacks can happen at the application layer as well as at network protocol layers. The DoS/DDoS defense mechanism must be build at both application and network protocol layers. For simplicity of the following discussion, we use the word DoS to represent DoS and DDoS.

Large-scale DoS attacks could potentially paralyze the Internet.

Examples of flooding based attacks comprise SYN flooding and ICMP flooding. Various mechanisms are developed for prevention, detection and response to DoS attacks. This is explained in particular in:

- "Denial of Service Attacks," CERT Coordination Center, http://www.cert.org/tech_tips/denial_of_service.html,
- Houle, K.J. and Weaver, G.M. "Trends in Denial of Service Attack Technology," CERT Coordination Center, October 2001, and
- "Distributed Denial of Service (DDoS) Attacks/tools," http://staff.washington.edu/dittrich/misc/ddos/.

These mechanisms together with other security means, such as firewall and intrusion detection system, provide managed security services to enterprises or ISP networks, which consists of routers, servers, and hosts.

Much research work have been done to defend or to mitigate DoS or DDoS attacks in the last decade as explained in particular in

- Chang, R.K.C., "Defending Against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial," IEEE Communication Magazine, October 2002,
- "Denial of Service Attacks," CERT Coordination Center, http://www.cert.org/tech_tips/denial_of_service.html,
- Houle, K.J. and Weaver, G.M. "Trends in Denial of Service Attack Technology," CERT Coordination Center, October 2001,

- Karig, D. and Lee, R., "Remote Denial of Service Attacks and Countermeasures," Princeton University Department of Electrical Engineering Technical Report CE-L2001-002, October 2001, http://www.princeton.edu/~rblee/ELE572Papers/karig01DoS.pdf,
- Khattab, S.M., etc. "Proactive Server Roaming for Mitigating Denial-of-Service Attacks," 2003, http://www.cs.pitt.edu/NETSEC/publications_files/itre03.pdf, and
- "Distributed Denial of Service (DDoS) Attacks/tools," http://staff.washington.edu/dittrich/misc/ddos/.

One basic approach is to detect DoS attack and to filter out attack packets. The Internet infrastructure is hierarchical. The attack detection and attack packets filtering can be done at different levels of the network, for example, at local computer, local network, local ISP network, or upstream ISP network. The effectiveness of the attack detection and packet filtering is dependent on the network level or levels that the defensive mechanisms are executed.

Existing research mainly focuses on protecting enterprise or ISP networks that consist of servers, routers, and host computers. Most approaches are complex; some involve changing Internet infrastructure routers and others require changing servers and clients. Very little literature is found to protect resource constrained embedded network devices from DoS attacks. Lee etc. proposed a port hopping method that can be used for embedded devices, as explained in Lee, H.C.J. and Thing, V.L.L., "Port Hopping for Resilient Networks," 2004, http://www.diadem-firewall.org/publications/VTC2004Fall_Port-Hopping.pdf. With the port hopping, the UDP/TCP port number used by the server varies as a function of time and a shared secret between the server and the client. The method simplifies the detection and filtering of malicious packets. It should work if a network device has agreed with its client or server on the port hopping mechanism. However, because this method requires both clients and their server implement the port hopping, it does not work if a network device provides a standard service, such as a web server, and allows access from a

standard client; or if a network device is a standard client, such as web browser, and is to access a standard server.

Many commercial security products, such as intrusion detection system (IDS) are available that can be placed in networks to secure and protect the networks. Internet Firewalls, global defense infrastructures, are used to protect Internet. These approaches protect the embedded network devices in the way that they protect the network. However, the embedded network devices are still in danger if the devices are connected to the Internet via host computers. A host computer may launch, knowingly or unknowingly, DoS attacks against connecting devices. For example, a maliciously installed malware or a computer worm on a host computer may launch such attacks. Because attack packets may not go to the outer network, defense mechanisms setup at the network level or at routers will not be able to help. The resource constrained network devices much have their own defense mechanisms that can live with the limitations of the devices.

Existing designs of the mechanisms are typically not suitable or not as effective for small resource constrained network devices because of the limited resources and bandwidth. Even the computer level DoS defending mechanisms may not be effective for embedded systems. For example, the SYN cookies approach developed for Linux avoids memory consumption for half-open connections. This prevents SYN flooding-based DoS attack from the computer memory perspective. However, it still takes CPU power to react to SYN messages. This may work fine for modern computers. For small resource constrained network devices, all CPU power may be used to respond to SYN messages. The applications on the device cannot get the CPU time. The attacker still achieves the goal of DoS attack. On the other hand, if there are no appropriate security measures for resource constrained network devices; users will be reluctant to connect their devices to the Internet due to security concerns.

Two common DoS prevention mechanisms have been devised, and tailored to resource-constrained network devices. Other prevention methods may also be added.

The first one is known as packet filtering, and is a network security method, which, as its name says, filters incoming or outgoing packets to let good packets pass and to block suspicious packets. Filter rules specify what packets to pass and what packets to reject, thus controlling the packet filtering behavior. Packet filtering helps to protect network devices from DoS attacks to a certain extent as it can filter out potential DoS attack packets.

A multi-stage packet filtering method for resource-constrained network devices is described in US 11/246,736. This method is used as a security measure as well as a memory management scheme to deal with the limited memory resource.

The second method is known as SYN Cookies. Currently Linux kernel and three BSD's (Open, Free, Net) include a facility called SYN cookies, which is used to prevent SYN flooding attack and was proposed in Bernstein, D.J., SYN Cookies, http://cr.yp.to/syncookies.html. Zuquete proposed an improvement to SYN cookies along with details of the Linux SYN cookies implementation, as explained in Zuquete, A., "Improving the Functionality of SYN Cookies," http://www.inesc-id.pt/pt/indicadores/Ficheiros/165.pdf. The improvement is significant if all TCP implementations follow TCP specification. Unfortunately, this is not the case. Therefore the success of the improved SYN cookies depends on TCP client implementations. Nevertheless, SYN cookies are suitable for resource-constrained network devices, especially when the device has hardware or efficient software implementation of a hash function.

One of the elements encoded in a SYN cookie represents the requesting TCP client's maximum segment size (MSS), which specifies the maximum segment size that the client can receive. The data is a 3-bit encoding of 8 predefined MSS values. The client's MSS is approximated by one of these 8 MSS values. The MSS information is encoded in a SYN cookie because the MSS option only appears in a TCP SYN segment. A resource-constrained network device has very limited memory resource. It is unlikely to

exceed its TCP client's MSS. Therefore, the MSS encoding in SYN cookies may not be needed. Based on the risk analysis in Zuquete, A., "Improving the Functionality of SYN Cookies," http://www.inesc-id.pt/pt/indicadores/Ficheiros/165.pdf, not including MSS in SYN cookies can reduce the probability of SYN guessing, which is another network-based attack.

The Linux SYN cookie includes client's ISN. However, the client's ISN may be random, which is the case, for example, Firefox web browser. The original reason for including client's ISN is for the cookie to increase at least as fast as the client's ISN. Because it is random, there is no need to include client's ISN.

The SYN cookies facility may be turned on dynamically. In normal operation, SYN cookies are not necessary. The network device may use SYN cookies when there is a suspicion of SYN flooding attack.

Methods have also been devised for network-Based Attack Monitoring and Detection. Although it is possible to make DoS attacks harder to carry out, quite often DoS attackers are still able to find ways to carry them out. Small resource constrained network devices are more vulnerable because they have very limited computing resource and bandwidth. When under attack, the device shall be able to detect DoS attacks.

In order to defend against network-based attacks, many attack detection or intrusion detections methods are developed. Most existing detection methods are based on statistics of the packets or signatures (or patterns) of packets flows as explained in particular in:

- Chang, R.K.C., "Defending Against Flooding-Based Distributed Denial-of-Service Attacks: A Tutorial," IEEE Communication Magazine, October 2002,
- A. Habib, M. Hefeeda, and B. Bhargava, "Detecting Service Violations and DoS Attacks," NDSS Conference Proceedings, Internet Society, 2003,
- Karig, D. and Lee, R., "Remote Denial of Service Attacks and Countermeasures," Princeton University Department of

Electrical Engineering Technical Report CE-L2001-002, October 2001.
http://www.princeton.edu/~rblee/ELE572Papers/karig01DoS.pdf,

- Porras, P. and Neumann, P. "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," National Information Security Conference, 1997, http://www.sdl.sri.com/projects/emerald/emerald-niss97.html,
- Siaterlis, C. and Maglaris, B., "Towards Multisensor Data Fusion for DoS Detection," The 19th Annual ACM Symposium on Applied Computing, March 2004, http://www.netmode.ntua.gr/papers/papers/siaterlis_sac04.pdf, and
- Valdes, A., Skinner, K., "Adaptive, Model-based Monitoring for Cyber Attack Detection," http://www.sdl.sri.com/papers/a/d/adaptbn/adaptbn.pdf.

These methods require the understanding of normal or abnormal packets probabilities or signatures and being able to separate normal and abnormal packet flows. The systems need to be trained and be adaptable to changes of the statistics or signatures. Most of these methods deal with known attacks well but have difficulties to unknown attacks. Resource-constrained network devices can only adopt some existing methods if they are not resource consuming nor computational expensive.

From the implementation perspective, most existing attack detection methods are implemented as network components. Research has shown that detection upstream in a network is more effective than detection down stream. On the other hand, detection in host computers is also important because network-based attacks may be in the local network. Existing designs of detection methods in host computers typically run as separated processes or separate tasks. This means the detection module runs all the time when it is turned on. A resource-constrained embedded network device cannot afford to use this kind of approach because the detection module would compete with the device's main task for the very limited computing source.

A detection method for network-based attacks, designed for small resource-constrained network devices, but also applicable to larger computer systems, has been proposed in Lu, H.K., "A Method of Detecting Network-based Attacks for Resource-Constrained Network Devices," PCT/IB2006/003650.

It is based on the system's operational behavior, instead of packets probabilities or signatures. The model expresses expected operational behaviors and attack-suspicious behaviors of the system. Real-time monitors raise an alarm when the system has deviated from the expected operations or when attack-suspicious behaviors occur.

This detection mechanism is embedded inside the network module. This provides real-time detection and conserves memory. The small monitoring code only runs when the code execution passes through there. Therefore, no separate task is required for the detection module and the impact on the system performance is minimum. Additional advantages of this approach include the ability to detect unknown attacks, simplicity, extensibility, and flexibility. The effectiveness of the detection does not depend on known packets probabilities or signatures. Furthermore, the method can be combined with other existing packets based approaches.

When a resource-constrained network device is under flooding-based DoS attack, it is busy reacting to data input (data-in) requests. It typically does not have enough CPU to do anything else, and the DoS attack succeeds. There's a known approach of server roaming described in

- Khattab, S.M., etc. "Proactive Server Roaming for Mitigating Denial-of-Service Attacks," 2003, http://www.cs.pitt.edu/NETSEC/publications_files/itre03.pdf, and
- Khattab, S.M., etc. "Roaming Honeypots for Mitigating Service-level Denial-of-Service Attacks," 24th International Conference on Distributed Computing Systems (ICDCS'04).

But this approach turns out not to be helpful because the network device here is most likely a personal device that cannot be replaced without strict procedures.

It is an aim of the invention to provide a framework for resource constrained network device, for further protecting resource constrained network devices from DoS attacks. The invention involves an interrupt-based method for the resource-constrained network devices to respond in particular to flooding-based DoS attacks. The framework according to preferred embodiments of the invention comprises methods for prevention, detection, self-protection, continuation, and mitigation.

According to the invention, the problem of better protecting a resource constrained network devices from DoS attacks is solved by having a resource constrained network device comprise detection means for detecting DOS attacks, and data-in interrupt means for notifying the device of network data input requests, and means to disable data-in interrupt means notifications when a DOS attack is detected. This is advantageous because during the DOS attack, the device is no longer overloaded with input requests, and can find the time to execute vital tasks, such as protection tasks etc.

The invention and its advantages will be explained more in details in the following specification referring to the appended drawings, in which:

Figure 1 represents a framework for protecting resource-constrained network devices from DoS attacks,

Figure 2 represents interactions among the components of the protection framework of Figure 1 and other software modules of a device according to a preferred embodiment of the invention,

Figure 3 represents operational states illustrating how a response mechanism according to the invention may find enough time to work properly by managing data input interrupts,

Figure 4 shows how a network module of a device according to a preferred embodiment of the invention may get some CPU time during a self-protection procedure,

Figure 5 shows some options offered to users of devices according to preferred embodiments of the invention.

According to a preferred embodiment of the invention, a framework for protecting resource constrained network devices from DoS attacks, consists of several components for prevention, detection, self-protection, continuation, and mitigation, as illustrated in Figure 1.

This protection framework interacts with software modules of the resource-constrained network device to protect the system, data, applications, and the services on the device. Figure 2 illustrates the interactions among components of the protection framework and other software modules of the device.

The protection module provides measures to prevent DoS attacks, such as packet filtering and TCP Sync cookie. The detection module monitors the whole systems operational behavior to detect possible DoS attacks. The module can also include traditional detection methods by monitoring the statistics or signatures of the packets. Once a possible attack is detected, the detection module informs the operating system, which invokes the self-protection procedure. The network stack may also try to continue legitimate communications to enable applications to continue network activities. The continuation is supported by managing I/O interrupts, active packet filtering and a mitigation method that tries to stop DoS attack.

Self-protection aims to protect the data and the applications on the network device. Continuation aims to continue the service if possible. Mitigation tries to stop the DoS attack.

According to a preferred embodiment, when an attack is detected and the alert is sent to the OS, the OS disables the hardware interrupt for data-in. This gives the system CPU to run the self-protection procedures, which are described below. Once such procedures finish, the OS enables the hardware interrupt for data-in. The corresponding interrupt handler and the network module can function again. Because the device is still under attack, the

network module actively filters out unwanted packets as early as possible and actively looks for expected packets for outstanding connections. Once an expected packet arrives, the hardware data-in interrupt is disabled again. The network module processes the packet, and may pass the data to the application. Once the packet is processed and consumed, the hardware data-in interrupt is enabled again, as illustrated on Figure 3. Because the TCP is a reliable transmission protocol and the device can process packet fast enough, missed packets will be limited. Even if packets were missed, they will be retransmitted again from the other end.

According to a preferred embodiment, once the network module of the resource-constrained network device detects a network-based attack, it alerts the operating system. It may voluntarily suspend its task to give the CPU to other applications. The operating system invokes self-protection procedures, which alert and schedule the tasks to protect data, files, and applications. The self-protection procedures include securing sensitive in-memory data to secure storage, finishing outstanding file transactions, saving future-needed application contexts, and ending certain tasks or applications.

As mentioned earlier, the operating system disables the data input interrupts to gain CPU time for the self-protection procedures. The operating system may choose to execute the self-protection procedures uninterrupted by disabling timer interrupts or to allow some interruptions by allowing timer interrupts. Even in the latter case, very limited timer handlers are enabled. In this case, the network module can set up a timer to get some CPU time. The timer interrupt handler temporarily enables the data-in interrupt to enable the network module to catch incoming packets. The network module drops all incoming packets, except the TCP packets with ACK set for previous sent messages. Such ACK packets free the send buffers. They also enable the network module to release CPU to applications tasks. For example, this enables the socket send() function to return back to its call, as illustrated on Figure 4. The network module should only do some very quick thing in this situation and return control to the operating system. The operating system disables the data-in interrupt to continue the self-protection procedures.

Once the self-protection procedures finish, the OS enables the hardware interrupt for data-in. The corresponding interrupt handler and the network module can function again. The network module finishes the queued task as much as it can, for example, send out queued out-going messages. If the network device is a server, it may send a message to its clients, which will be described below.

The network module preferably continues dropping all incoming packets as early as possible, for example, at the interrupt level. The network module continues to filter out unwanted packets, performs its network stack work and enables the applications to work, which is described below.

Using the detection and self-protection methods described earlier, when a resource-constrained network device is under a network-based attack, it is still alive. During and after executing self-protection procedures, the network module is given some CPU time and may finish sending pending messages. The device might not miss much useful packets even though the incoming packets were dropped. The Internet server or client that the device was connected with will resend the messages, if the connections are not timed out yet, because the TCP is a reliable transmission protocol. At this time, there are several options for continuation, as illustrated on Figure 5. The network device can inform the user these options via sending a message, for example, sending a web page to the browser that the user is using.

The message can warn the user that the device is under a DoS attack and provide options and instructions to the user. The user can choose to disconnect the device or to continue to use the device. If he chooses to disconnect the device, he does so, for example, by taking out his network device, and tries again some other time or uses a different computer. If he chooses to continue using the device, he has several options: continue with the current session, start a new session, or finish the current session and then start a new session. There are, at least, two ways to start a new session. The

continuation method is described below. The new session method is described further below.

When the network device finished the safety procedure as explained above, the network module of the device can start to process the packets again. If the DoS attack has stopped, the device goes back to the normal operation. If the attack is still going on, the network module drops all incoming packets except those from the Internet client or server that the network device is communicating with. With the detection system's feedback, the front-end filter can filter out offending packets. For example, if SYN flooding attack is detected, all SYN packets may be filtered out, except those expected ones. Practically, this is a control system.

During this continuation, the data-in interrupt is turned on and off as explained earlier, and as illustrated in Figure 4. The device might be slow, but it should continue to function. The warning message to the user preferably informs the user that the operation might be slow. When the current session is finished, the device preferably provides the user an opportunity to start a new session.

How to start a new communication session in order to mitigate the DoS attack is described below. The method may stop the DoS attack.

Resource constrained network devices may be implemented in different ways. In most cases, a device can choose its own IP address. A device might even have more than one IP address. From the host perspective, the device is a network. In these cases, when the network device is under a DoS attack and the user has decided to start a new communication session, the device can enable IP hopping, as explained in particular in Jones, J., "Distributed denial of service attacks: Defenses, a special publication," Global Integrity, Technical Report, 2000. It basically discards its current IP address and assigns itself a new IP address. If the DoS attack is a targeted attack, for example, SYN flooding, the device can simple drop the messages for the old IP address initially. Once the host has learned the new IP address and updated its ARP cache and its routing table, the packets for the old IP

address will not come to the device because the device has a different IP address now. This stops the DoS attack. In order for this to work, the invention solves the two following problems:

(1) how to inform the user of the new IP address

(2) how to inform the host of the new IP address

The network device provides a web server and the user interacts with his device through a standard web browser. As mentioned earlier, when under a DoS attack, the device sends a warning and instruction web page to the browser. We then assume that the user has chosen to start a new session. According to the invention, there are at least two ways for the user to start a new session:

(1) The warning and instruction web page contains an http (or https) link that points to the new URL (with the new IP address) of the device. The user can click the link to start a new session.

(2) Some USB network devices, such as the Network Smart Card, allow the user to get to the login web page of the device by click an icon through the corresponding mass storage interface. In this case, the user closes the current browser, goes back to the device's mass storage device interface, and clicks the icon to start a new session.

In both methods, the user does not need to know the new IP address of the network device. The device either sets the link in the warning and instruction web page, in method 1, or changes its address in the startup file that is clicked by the user from the mass storage device interface, in method 2. The browser will use the new address specified in the link or in the startup file to open the device's login page.

During the process, the host will send an ARP asking who has this IP address. The network device will respond. The host will automatically update its ARP cache and routing table to reflect the device's IP address. The device can now function using the new IP address.

The old IP address of the network device will be eventually dropped out from the host's ARP cache and routing table. The flooding packets of the DoS attack will then no longer be sent to the device. This technique thus stops the DoS attack. In our experiments, we used a Windows XP laptop as a host

computer to run a web browser to access the web server in the network device. Two or three minutes after the network device changed its IP address, the old IP address was dropped out from the host's ARP cache. This was observed from both "arp –a" shell command and from a protocol analyzer ethereal.

The approach described here has advantages over existing IP hopping, which protect a public server, whose clients use Domain Name Server (DNS) to lookup the server's IP address. Before the DNS updates its cache for the IP address change, clients' messages may be filtered out by a network firewall. The clients cannot reach the server during this period. Therefore, the IP hopping has latencies. For the method proposed here, the user is actively involved in the process to make decisions and to make the transition happen directly. The actual new IP address is used to make a new connection instead of through DNS. In addition, the network device, such as a network smart card, may be clients as well as servers, both of which can take advantage of IP hopping.

CLAIMS

1. A resource constrained network device comprising detection means for detecting DOS attacks, and data-in interrupt means for notifying the device of network data input requests, characterized in that it comprises means to disable data-in interrupts when a DOS attack is detected.
2. The resource constrained network device according to claim 1, comprising self protection means for protecting the contents of said device in case of DOS attacks.
3. The resource constrained network device according to claim 1, comprising continuation means for continuing operation of said device despite the DOS attack.
4. The resource constrained network device according to claim 1, comprising mitigation means set to stop the current communication session and to start a new communication session in order to escape the current DOS attack.
5. The resource constrained network device according to claim 2, wherein said device comprises an operating system, said operating system being set to enable access to data stored in said device, the self protection means being set to notify the operating system of the attack in order to protect said data.
6. The resource constrained network device according to claim 5, wherein said device comprises memory and secure storage, wherein the self protection means comprise means for securing sensitive data located in memory into secure storage.
7. The resource constrained network device according to claim 5, wherein the operating system is set to manage a file system, and wherein the self protection means comprise means for finishing outstanding file transactions.
8. The resource constrained network device according to claim 2, wherein said device comprises an operating system, said operating system being set to enable access to applications stored in said device, the self

protection means being set to notify the operating system of the attack in order to protect said applications.

9. The resource constrained network device according to claim 8, wherein the self protection means comprise means for ending certain tasks or applications.

10. The resource constrained network device according to claim 8, wherein the self protection means comprise means for saving application contexts which may be needed in the future.

11. The resource constrained network device according to claim 2, wherein after self protection of the device against a DOS attack by self protection means, the data-in interrupts are enabled again.

12. The resource constrained network device according to claim 1, comprising timer interrupt means for notifying the device at times defined by a timer, and comprising means to enable timer interrupts when a DOS attack is detected.

13. The resource constrained network device according to claim 12, wherein the device is set to set up the timer in order to allocate only a small fraction of the CPU time for processing network data input requests while a DOS attack is underway.

14. The resource constrained network device according to claim 1 or 13, comprising means to temporarily enable data-in interrupts after a DOS attack is detected in order to enable the processing of some input data.

15. The resource constrained network device according to claim 14, wherein the device uses the TCP/IP protocol, and wherein upon detection of a DOS attack, the device is set to drop all network data input requests, except those corresponding to TCP packets with ACK set for previous sent messages, in order to free the send buffers.

16. The resource constrained network device according to claim 1, wherein upon detection of a DOS attack the device is set to send a message to the user of said device in order to inform him of the DOS attack.

17. The resource constrained network device according to claim 1, wherein said device comprises a server, and wherein upon detection of a DOS attack, the device is set to send a message to clients connected to said server.

18. The resource constrained network device according to claim 16 or 17, wherein the message consists of a web page designed to be displayed to the user of said device in order to inform him of the DOS attack.

19. The resource constrained network device according to claim 3, wherein upon DOS attack detection the continuation means are set to continue with the current session, or to start a new session, or to finish the current session and then start a new session.

20. The resource constrained network device according to claims 16 and 19, wherein the message sent to the user contains a request to select between at least two of the following options:

    a. continuing with the current session,.or

    b. starting a new session, or

    c. finishing the current session and then starting a new session,

    and wherein the continuation means are set to proceed according to the selection.

21. The resource constrained network device according to claim 20, wherein the message comprises a link associated with each option, and wherein the selection is done by clicking the relevant link.

22. The resource constrained network device according to claim 4, wherein the device uses the IP protocol, and wherein upon DOS attack detection the mitigation means are set to discard the current IP address of said device and to assign a new IP address.

23. The resource constrained network device according to claims 16 and 22, wherein the message sent to the user contains the new IP address of said device.

24. The resource constrained network device according to claim 23, wherein said new IP address is associated with a link, wherein said link is set to be displayed in the message, and wherein clicking said link triggers a new communication session with said device using the new IP address.

25. The resource constrained network device according to claim 22, wherein said device is a USB device comprising a mass storage interface for accessing said device from a host, and wherein upon connection to said device through the mass storage interface, the mass storage interface is

set to initiate a new communication session with said device using the new IP address.

26. The resource constrained network device according to any previous claim, wherein said device is a smart card.

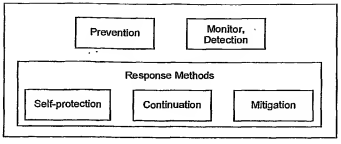


Figure 1

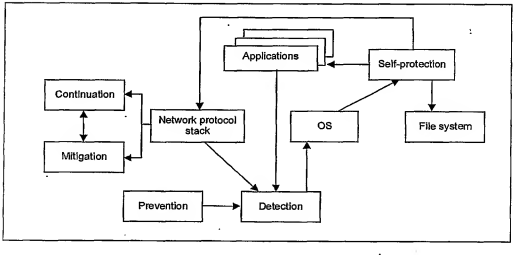


Figure 2

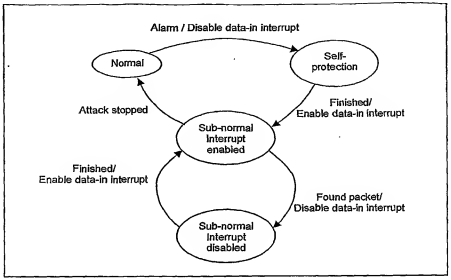


Figure 3

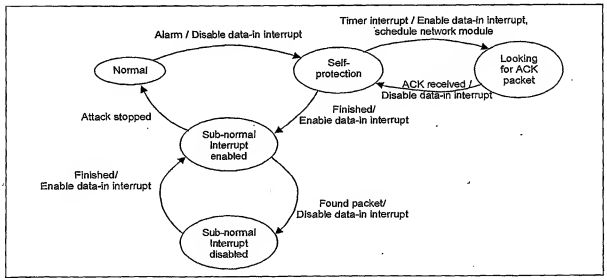


Figure 4

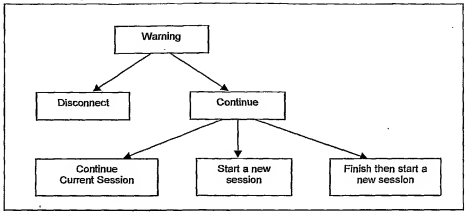

Warning
Disconnect
Continue
Continue Current Session
Start a new session
Finish then start a new session

Figure 5

DERWENT-ACC-NO: 2007-816533

DERWENT-WEEK: 200807

TITLE: Resource constrained network device e.g. network smart card, for e.g. intrusion detection system, has detection module monitoring system`s operational behavior, and operating system set to enable access to data

INVENTOR: LU H K

PATENT-ASSIGNEE: AXALTO SA[AXALN]

PRIORITY-DATA: 2006US-793934P (April 21, 2006)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE
WO 2007122495 A2	November 1, 2007	EN
WO 2007122495 A3	January 17, 2008	EN

DESIGNATED-STATES: AE AG AL AM AT AU AZ BA BB BG BH BR BW BY BZ CA CH CN CO CR CU CZ DE DK DM DZ EC EE EG ES FI GB GD GE GH GM GT HN HR HU ID IL IN IS JP KE KG KM KN KP KR KZ LA LC LK LR LS LT LU LY MA MD MG MK MN MW MX MY MZ NA N G NI NO NZ OM PG PH PL PT RO RS RU SC SD SE SG SK SL SM SV SY TJ TM TN TR TT TZ UA UG US UZ VC VN ZA ZM ZW AT BE BG BW CH CY CZ DE DK EA EE ES FI FR GB GH GM GR HU IE IS IT KE LS LT LU LV MC MT MW MZ

NA NL OA PL PT RO SD SE SI SK SL SZ TR TZ UG ZM ZW AE A G AL AM AT AU AZ BA BB BG BH BR BW BY BZ CA CH CN CO CR CU CZ DE DK DM DZ EC EE EG ES FI GB GD GE GH GM GT HN HR HU ID IL IN IS JP KE KG KM KN KP KR KZ LA LC LK LR LS LT LU LY MA MD MG MK MN MW MX MY MZ NA NG NI NO NZ OM PG PH PL PT RO RS RU SC SD SE SG S K SL SM SV SY TJ TM TN TR TT TZ UA UG US UZ VC VN ZA ZM ZW AT BE BG BW CH CY CZ DE DK EA EE ES FI FR GB GH GM GR HU IE IS IT KE LS LT LU LV MC MT MW MZ NA NL OA PL PT RO SD SE SI SK SL SZ TR TZ UG ZM ZW

APPLICATION-DATA:

PUB-NO	APPL-DESCRIPTOR	APPL-NO	APPL-DATE
WO2007122495A2	N/A	2007WO-IB001052	April 23, 2007

INT-CL-CURRENT:

TYPE	IPC DATE
CIPP	H04L29/06 20060101
CIPS	G06F21/00 20060101
CIPS	G06F9/48 20060101

ABSTRACTED-PUB-NO: WO 2007122495 A2

BASIC-ABSTRACT:

NOVELTY - The device has a detection module monitoring a system`s operational behavior to detect possible denial of service (DoS) attacks. An operating system is set to enable access to data stored in the device. A network module continues dropping all incoming packets at an interrupt level. The network module continues to filter out unwanted packets. The operating system disables data input interrupts to gain CPU time for self-protection procedures.

USE - Used for a commercial security product such as intrusion detection system.

ADVANTAGE - The operating system protects resource constrained network devices from denial of service (DoS) attacks. The device is not overloaded with input requests, and finds the time to execute vital tasks such as protection tasks.

DESCRIPTION OF DRAWING(S) - The drawing shows an operational state illustrating how a response mechanism finds enough time to work properly by managing data input interrupts.

CHOSEN-DRAWING: Dwg.3/5

TITLE-TERMS: RESOURCE CONSTRAIN NETWORK DEVICE SMART CARD INTRUDE DETECT SYSTEM MODULE MONITOR OPERATE BEHAVE SET ENABLE ACCESS DATA

DERWENT-CLASS: T01 T04

EPI-CODES: T01-F02A1; T01-F05G; T01-H07A; T01-J12; T01-N01A2C; T01-N02B1D; T01-N02B1E; T01-N02B2B; T04-K03A;

SECONDARY-ACC-NO:

Non-CPI Secondary Accession Numbers: 2007-649233

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
17 November 2005 (17.11.2005)

PCT

(10) International Publication Number
WO 2005/109824 A1

(51) International Patent Classification?: H04L 29/06

(21) International Application Number:
PCT/US2005/011702

(22) International Filing Date: 5 April 2005 (05.04.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/832,588 27 April 2004 (27.04.2004) US

(71) Applicant *(for all designated States except US)*: CISCO TECHNOLOGY, INC. [US/US]; 170 West Tasman Drive, San Jose, CA 95134-1706 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: ITHAL, Ravishankar, Ganesh [IN/US]; 1401 Red Hawk Circle, Apartment E106, Fremont, CA 94538 (US).

(74) Agents: HUANG, David, E. et al.; Chapin & Huang, LLC, Westborough Office Park, 1700 West Park Drive, Westborough, MA 01581 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

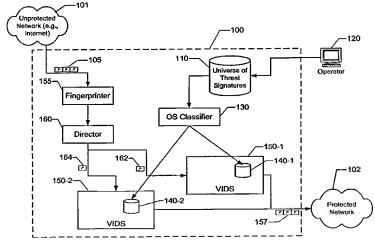
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii)) for the following designations AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NI, NO, NZ, OM,*

[Continued on next page]

(54) Title: SOURCE/DESTINATION OPERATING SYSTEM TYPE-BASED IDS VIRTUALIZATION



(57) Abstract: Systems and methods for virtualizing network intrusion detection system (IDS) functions based on each packet's source and/or destination host computer operating system (OS) type and characteristics are described. Virtualization is accomplished by fingerprinting each packet to determine the packet's target OS and then vetting each packet in a virtual IDS against a reduced set of threat signatures specific to the target OS. Each virtual IDS, whether operating on a separate computer or operating as a logically distinct process or separate thread running on a single computer processor, may also operate in parallel with other virtual IDS processes. IDS processing efficiency and speed are greatly increased by the fact that a much smaller subset of threat signature universe is used for each OS-specific packet threat vetting operation.

WO 2005/109824 A1

PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VC, VN, YU, ZA, ZM, ZW, ARIPO patent (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG)

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii)) for all designations*

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

SOURCE/DESTINATION OPERATING SYSTEM TYPE-BASED IDS VIRTUALIZATION

BACKGROUND

A typical computer networking system may include, among other things, an intrusion detection system (IDS) configured to monitor network traffic and to block attempted attacks on or intrusions into the protected network space. Such intrusion detection systems may include or coexist with various types of firewalls, packet monitors, and other devices that typically include intrusion sensing functions (e.g., advanced routers). These systems include both active and passive devices and are generally referred to as "sensors."

An IDS may include, among other things, a network interface for receiving packets, a packet filtering mechanism for determining whether or not to accept inbound packets, memory for storing threat signatures, and a network interface for transmitting (or forwarding) packets into the protected network. The aforementioned elements of an IDS may be implemented in either hardware or software or some combination of both.

An IDS sensor may also be virtualized. "Virtualized," as this term is used in the art, refers to virtualization, the practice of dividing the IDS functionality among multiple processes or logical elements each configured to operate in parallel with the others. These processes may run on separate processors (i.e., in separate pieces of hardware) or may run on a single processor in multiple threads. In this sense, virtualization may be thought of as another form of distributed processing: the functional or logical elements of the required process may be carried out in multiple locations, where "location" is understood as referring to both physical as well as logical separation. One of ordinary skill in the art will also recognize that virtualization does not necessarily require the division of IDS functionality into separate threads; the "feel" of providing multiple logical functions within a single physical device is all that is needed.

One conventional approach to IDS virtualization is to use to separate functionality (or to "virtualize") based on user-configured selection criteria such as packet IP prefix, domain name, VLAN, input interface, etc. In order to provide the same level of threat protection in each virtual IDS (or virtual IDS process), however, packets destined to be processed by a given virtual IDS sensor must be checked against signatures of vulnerabilities for all

operating systems known to exist on the unprotected network. For typical cases where the unprotected network consists of the public Internet, this universe of operating systems is the universe of all known operating systems. Likewise, when the protected network is sufficiently large and diverse, the vulnerabilities in the destination host must include the vulnerabilities of all known operating systems as well.

In typical IDS systems, known vulnerabilities are stored in IDS memory as threat signatures, i.e., descriptive information formatted so that it may be rapidly compared by the IDS to packet content in order to directly determined whether each particular threat is or is not present in the packet. As the number of threats grows, so too must the set or universe of threat signatures. As each new threat is identified, any aspect or manifestation of that threat not common to a previously seen threat signature necessitates the definition of a new threat signature.

SUMMARY

There are several notable deficiencies to the above-described conventional approaches. For example, the prior art systems do not generally scale well as the number of threat signatures continues to increase. In addition, the packet latency introduced by comparing (or "vetting") each incoming packet against the universe of threat signatures causes a significant degradation in performance. In general, the vetting of each packet against each threat signature in the signature universe is slower and more inefficient than is desirable in modern high-speed, high throughput packet processing IDS sensors.

In contrast to the above-described conventional approaches, embodiments of the invention are directed to systems and methods for virtualizing IDS functions based on the operating system (OS) characteristics of each packet's source and/or destination host computers. This virtualization is accomplished by fingerprinting each packet to determine the packet's target OS and then performing a rapid packet vetting against a reduced set of threat signatures appropriate to that target OS. The "target OS" may be either the operating system of the packet's source host or the operating system of the packet's destination host. In some embodiments, information about either or both OSs may be used to select a reduced set of threat signatures.

The vetting process for each packet proceeds by comparing aspects of the packet such as, but not limited to, packet data payload, header flags, options, source IP address . destination IP address, source port and/or destination port to the reduced threat signature set appropriate to the target OS for each particular packet.

Accordingly, each virtual IDS process, whether operating on a separate machine (computer), operating as a distinct thread running on a single computer or processor, or merely a logical distinction in functionality is thus able to operate in parallel with other virtual IDS processes, although parallel operation is not necessary to practice the present invention. In such an embodiment, each VIDS process performs packet vetting operations using the reduced threat signature set appropriate to the target OS as determined by the contents of each packet. IDS processing efficiency is greatly increased by the fact that a much smaller subset of threat signature universe is used for each packet threat vetting operation. Furthermore, as the universe of threat signatures increases without bound (as it is currently expected to do), IDS efficiency will not be greatly impacted since packets not subject to new vulnerabilities will not have to be vetted against those new threats. In fact, processing efficiency may be greatly enhanced by the ability of embodiments of the invention to greatly reduce the set of threat signatures that need to be searched for each packet. For example, it is well-known in the art that Apple Computer's OS X operating system is not vulnerable to the vast majority of attacks seen "in the wild" (i.e., known to be loose) today. Packets destined for this OS thus do not need to be vetted against threats to the Microsoft Windows OS, for example. Since the number of threat signatures is so much smaller, processing (including threat vetting) speed for OS X-destined packets is greatly increased.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, features and advantages of the invention will be apparent from the following description of particular embodiments of the invention, as illustrated in the accompanying drawings in which like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the invention.

Figure 1 is a high-level block diagram of an intrusion detection system configured according to one embodiment of the present invention.

Figure 2 is a high-level flowchart of the process of IDS virtualization, according to one embodiment of the present invention.

DETAILED DESCRIPTION

Embodiments of the present system are directed to techniques for intrusion detection system (IDS) virtualization based on packet target operating system (OS) characteristics and the tailoring of packet vetting to a reduced threat signature set. Tailoring, and the concomitant IDS virtualization, is based in turn on the use of passive and/or active packet fingerprinting to determine the packet's target operating system. The target OS may be, in some embodiments, a tuple consisting of the operating system of the packet's source host and the packet's destination host.

Figure 1 illustrates a virtualized intrusion detection system 100 configured according to one embodiment of the present invention. Viewed at a high level, system 100 consists of an interface to an unprotected network 101, the vetting/processing system components, and a second interface to a protected network 102. Additionally, an operator interface 120 allows an operator to load the universe of threat signatures.

Within the vetting/processing components of system 100, packets are processed by a fingerprinter, which in turn supplies fingerprinted packets to a director for switching or routing to one or more virtual IDS (VIDS) processes (which, as noted above, may be logically distinct in any of several ways). After vetting (with the aid of reduced threat signature sets 140) in the appropriate VIDS process (or set of VIDS processes), packets leave system 100 via the second network interface and enter protected network 102.

In one embodiment of the invention in particular, a stream of packets 105 from an unprotected network (such as, but not limited to, the public Internet) enters system 100 at fingerprinter 155 via a first network interface (not shown). Fingerprinter 155 looks at each packet to determine both the host operating system that sent the packet as well as the operating system of the host computer for which the packet is destined. This fingerprinting process, carried out on each packet in packet stream 105, is accomplished using either passive or active fingerprinting methods and techniques commonly used and well-known in the art. Passive fingerprinting techniques are discussed in detail in Toby Miller, Passive OS Fingerprinting: Details and Techniques, available at http://www.sans.org/rr/-

special/passiveos.php (last viewed 4/13/04), and Passive OS Fingerprinting: Details and Techniques (Part 2), available at http://www.sans.org/rr/special/passiveos2.php (last viewed 4/13/04), both incorporated herein by reference in their entireties. Active fingerprinting techniques are discussed in detail in (for example) Fyodor, Remote OS detection via TCP/IP Stack Fingerprinting, available at http://www.insecure.org/nmap/nmap-fingerprinting-article.html (last viewed 4/13/04), and Ofir Arkin, ICMP Usage In Scanning, Version 3.0, available at http://www.sys-security.com/archive/papers/ICMP_Scanning_v3.0.pdf (last viewed 4/13/04), both of which are incorporated herein by reference in their entireties. As the design and configuration of the hardware and/or software necessary to carry out either passive or active fingerprinting (or both) is well within the skill of an ordinary practitioner, implementation of specific fingerprinting techniques is not further discussed herein.

As each packet is fingerprinted, information denoting the source and destination operating systems is associated with the packet. This information may be stored, for example, in a temporary or scratchpad memory for use by director 160. Alternatively, target operating system data may be appended to or concatenated with the packet as it is passed to director 160.

In many implementations, the destination operating system (i.e., the operating system of the destination host) is of most use in evaluating the threat posed by packet; the "target OS" is the destination host OS because the most common threat is a threat to the packet's destination. In some cases, however, certain threat signatures are defined in terms of the source operating system, i.e., the "target OS" is the operating system of the host that sent the packet. For example, if the source operating system can be identified through fingerprinting as UNIX, then the fact that the packet contains a Microsoft Windows remote procedure call (RPC) indicates that a UNIX box may be attempting to penetrate a Windows system. Since a Windows RPC call would not normally come from a UNIX machine, it should be identified as a threat. Thus, some signatures may reference the source host OS as the target operating system.

Packets leaving fingerprinter 155 enter director 160 where, based on the target OS identified by fingerprinter 155, they are sent (i.e., redirected, switched, or routed, as those terms are known the art) to the appropriate virtual IDS (VIDS) process 150.

Although fingerprinter 155 and director 160 are depicted in Fig. 1 as separate functional elements, one of ordinary skill in the art will appreciate that these functions may be implemented in one or more devices and/or one or more to software processes. Accordingly, the implementation of virtualized IDS 100 is not limited to a particular distinction between or arrangement of hardware and/or software functionality and physical embodiment.

For the sake of clarity, Fig. 1 illustrates only two VIDS processes, 150-1 and 150-2, although in a typical implementation there may be many VIDS processes or units, each corresponding to a different target operating system. And, although each VIDS 150 is described as a process executing in software on a processor circuit, those skilled in the art will realize that the functions of a VIDS process may also be implemented in hardware (i.e., in a processor or other computer unit), in a single software process, or in a combination of hardware and software. Accordingly, the present invention is not limited to any particular implementation of the VIDS functionality.

By way of illustration, but not of limitation, VIDS 150-1 may be configured (in some embodiments of the invention) to compare individual packets 162 to a particular reduced threat signature set 140-1, corresponding to threats targeted at (for example) Windows XP. A second VIDS, 150-2, may be configured to compare packets 164 (which are directed to VIDS 150-2 because they share a target operating system that is different from that shared by packets 162) to a second reduced threat signature set 140-2. Reduced threat signature set 140-2 may list threats targeted at (for example) Windows 98. One of ordinary skill of the art will appreciate that many VIDS units 150 may be employed, each containing a reduced threat signature set 140 containing threat signatures appropriate to a different target operating system. Accordingly, the present invention is not limited to a particular number of VIDS units 150 and/or reduced threat signature sets 140.

Reduced threat signature sets 140 are formed by OS classifier 130 from a database or other memory or storage device 110 containing the universe of all known threat signatures. The universe of threat signatures may be loaded into database 110 prior to activation of the virtualized IDS 100 by an operator using a conventional workstation or computer 120. Alternatively, although not shown, the universe of threat signatures 110 may be loaded by file transfer, scan, self-discovery or monitoring of network traffic, or any of the conventional

means known in the art or yet to be discovered for creating and maintaining a database of threat signatures.

The process of forming discrete, reduced threat signature sets 140 from the universe of threat signatures 110 is accomplished by selecting signatures according to target operating system through conventional sorting methods and techniques. The threat signatures themselves are conventional text and/or digital data strings known and used in the art for packet-based threat vetting. The use and organization of such threat signatures are described in, for example, Kyle Haugsness, Intrusion Detection In Depth: GCIA Practical Assignment Version 3.0, (December 2, 2001), available at http://www.sans.org/rr/papers/23/835.pdf (last viewed 4/13/04); Syed Yasir Abbas, Introducing Multi Threaded Solution to Enhance the Efficiency of Snort, (December 7, 2002), available at http://www.cs.fsu.edu/-research/reports/TR-021204.pdf (last viewed 4/14/04); and the Snort Users Manual, v. 2.1.2, available at http://www.snort.org/docs/snort_manual/, last viewed 4/23/04), all of which are incorporated herein by reference in their entireties.

The actions of operator 120 and OS classifier 130 required to load threat universe database 110 and to select or form the various reduced threat signature sets 140 may be accomplished at any time prior to activation of virtualized IDS 100 or even during operation, as when certain threat signatures must be updated without interrupting packet processing in virtualized IDS 100. Accordingly, although reduced threat signature sets 140 must be initially defined, those definitions need not be static and the present invention is not so limited.

Certain threats are known to be common to more than one target OS or platform. Accordingly, threat signatures representing these common threats are grouped into a separate reduced threat signature set 140 (not shown) that is necessarily applied (in another VIDS 150) to all packets 105. This parallel processing capability may be implemented (in some embodiments of the invention) in director 160 by providing the capability to direct packets to more than one VIDS 150 at the same (or substantially the same) time. This capability for parallel (near-simultaneous) processing of packets by more than one VIDS 150 provides a significant speed and throughput increase in virtualized IDS 100.

Packets 162 or 164 (in the two-VIDS 150 embodiment of Fig. 1) that match a threat signature are blocked within VIDS 150 and not allowed to pass through into protected

network 102. Blocking may take any conventional form, such as but not limited to deletion, routing to /dev/null, packet marking and routing to a special process for alarming/reporting, or some other means of preventing the threatening packet from leaving virtualized IDS 100. Conversely, packets that do not match any threat signature form a vetted (or accepted) packet stream 157, which is then passed on (through conventional means) into protected network 102. Protected network 102 may consist of, for example but not by way of limitation, an intranet, LAN, MAN, or other relatively-closed network space secured and protected by virtualized IDS 100.

Figure 2 illustrates one exemplary embodiment of a process 200 whereby the virtualized IDS receives packets, performs fingerprinting and packet redirection to an operating system-appropriate VIDS process, compares each packet to a specific reduced threat signature set, and passes or rejects packets accordingly.

Process 200 begins prior to virtualized IDS packet processing operations with the loading of threat signatures in step 210. Step 210 defines (or collects) the universe of relevant threat signatures in a database or other memory or storage element. The process of signature definition and/or collection (including editing and/or revising as required) may be performed, in some embodiments of the invention, by conventional techniques and processes well-known in the art. Process 200 next forms from the threat signature universe a number of reduced threat signature sets, based on target operating system, in step 220. As noted above, each threat signature is specific to a target operating system, i.e., the threats themselves are generally targeted at only a single operating system. For threat signatures that apply to multiple operating systems (for example, vulnerabilities that exist in more than one operating system), a reduced threat signature set is formed for threats common to two or more operating systems. Each reduced set of threat signatures is stored in a database or other memory structure using conventional processes.

Each database (or set) of operating system-specific reduced threat signatures is then used, step 230, to form a virtualized IDS process configured to compare or vet each packet presented to it. These packets begin to arrive at comparing step 280 once the virtualized IDS begins to process incoming packets in step 250.

The virtualized IDS receives packets 250 and fingerprints each one in step 260. Fingerprinting 260 may be an active process, such as by querying a DHCP server or by other

active fingerprinting methods known in the art. Alternatively, fingerprinting 260 may be accomplished by passive means, such as (but not limited to) an analysis of the packet header's ACK, Flags, and/or Options fields and comparison of the values thereof to a set of OS-specific indicators. These indicators (or rules) permit the inference of the packet source host's and/or the packet destination host's operating system from the TCP and/or IP packet header field values.

After fingerprinting step 260 has determined the identity of the target OS for each packet, the packet is directed (in step 270) to a particular one of the several VIDS processes according to the packet's target OS. As discussed above, multiple VIDS processes may be created (in step 230) to match the number of distinct reduced threat signature sets formed in step 220.

In some implementations (not shown in Fig. 2), the process 200 checks whether the protocol carried by each packet can be officially "talked" (used, communicated, sent) by the source OS by further directing the packet (in step 270) to a second VIDS process for evaluating the source OS. The source OS VIDS checks only if the protocol is valid for the source OS, as further described below. A valid protocol passes the VIDS comparison for the source OS.

Within each VIDS process 275, each input packet is compared, step 280, to the corresponding reduced threat signature set for the packet's target OS. If the packet passes the comparison test, i.e., the packet does not match any threat signature in the reduced threat signature set, it is accepted for further processing and/or forwarding out of the VIDS process 275 in step 290. The VIDS process 275 then continues or loops back to comparison step 280 for the next packet presented.

If, on the other hand, the packet matches one of the threat signatures in the reduced threat signature set, the packet is dropped in step 299. "Dropped," in this context, refers to any of the various ways in which a bad packet may be processed in the intrusion detection context: the system may alternatively set an alarm, flag, reject, or null-route the packet, or otherwise suppresses its further transmission from the virtualized IDS.

It must be noted, although not depicted in Fig. 2 due to the need for clarity and simplicity in the drawing, multiple VIDS processes 275 (as represented by step 280, 290 and

299) may operate in parallel in some embodiments of the invention, thereby processing multiple packets at substantially the same time. In addition, more than one VIDS process 275 may process the same packet, as when both an OS-specific threat signature set and a common threat signature set are to be applied. Regardless of whether parallel processing is employed, however, the reduction in the threat signature set applied to each packet provides a significant performance benefit.

Alternate Embodiments

The order in which the steps of the present method are performed is purely illustrative in nature. In fact, the steps can be performed in any order or in parallel, unless otherwise indicated by the present disclosure.

The method of the present invention may be performed in hardware, software, or any combination thereof, as those terms are currently known in the art. In particular, the present method may be carried out by software, firmware, or microcode operating on a computer or computers of any type. Additionally, software embodying the present invention may comprise computer instructions in any form (e.g., source code, object code, interpreted code, etc.) stored in any computer-readable medium (e.g., ROM, RAM, magnetic media, punched tape or card, compact disc (CD) in any form, DVD, etc.). Furthermore, such software may also be in the form of a computer data signal embodied in a carrier wave, such as that found within the well-known Web pages transferred among devices connected to the Internet. Accordingly, the present invention is not limited to any particular platform, unless specifically stated otherwise in the present disclosure.

While this invention has been particularly shown and described with references to embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the invention as defined by the appended claims.

CLAIMS

I claim:

1. A method of intrusion detection system virtualization, comprising:
receiving a stream of packets;
fingerprinting each packet in a said stream to identify at least one target operating system (OS) type;
directing each said packet to a virtual IDS process corresponding to each said identified target OS type;
comparing each said packet to a threat signature set corresponding to each said identified target OS type in said virtual IDS process; and
accepting each said packet based on said comparing.

2. The method of Claim 1, wherein said fingerprinting comprises active fingerprinting.

3. The method of Claim 1, wherein said fingerprinting comprises passive fingerprinting.

4. The method of Claim 1, wherein said target OS type is the packet source host OS.

5. The method of Claim 1, wherein said target OS type is the packet destination host OS.

6. The method of Claim 1, wherein said target OS type is both the packet source host OS and the packet destination host OS.

7. The method of Claim 1, wherein, when said fingerprinting identifies more than one target OS type, said directing occurs at substantially the same time.

8. A method of intrusion detection system virtualization using operating system type information, comprising:
forming a plurality of reduced threat signature sets from a signature universe based on operating system type;
virtualizing an intrusion detection system (IDS) into a plurality of virtual IDS processes corresponding to said plurality of reduced threat signature sets;

receiving a stream of packets;

fingerprinting each packet in a said stream to identify at least one target operating system type;

directing each said packet to the virtual IDS process corresponding to each said target operating system type;

comparing each said packet to said reduced threat signature set in said virtual IDS process; and

accepting each said packet based on said comparing.

9. The method of Claim 8, wherein said plurality of reduced threat signature sets comprises a set of signatures common to more than one operating system type.

10. The method of Claim 8, wherein said fingerprinting comprises active fingerprinting.

11. The method of Claim 8, wherein said fingerprinting comprises passive fingerprinting.

12. The method of Claim 8, wherein said target operating system type is the packet source host operating system.

13. The method of Claim 8, wherein said target operating system type is the packet destination host operating system.

14. The method of Claim 8, wherein said target OS type is both the packet source host OS and the packet destination host OS.

15. The method of Claim 8, wherein, when said fingerprinting identifies more than one target operating system type, said directing to the virtual IDS process corresponding to each said target operating system type occurs at substantially the same time.

16. An apparatus for intrusion detection system (IDS) virtualization, comprising:

a first network interface connected to an unprotected network;

a fingerprinter operably connected to said first network interface for receiving a plurality of packets and configured to determine at least one corresponding target operating system (OS) fingerprint for each said packet;

a director connected to said fingerprinter configured to receive said plurality of packets and said corresponding target OS fingerprints, wherein said director directs each said packet to one or more virtual IDS units according to said at least one corresponding OS fingerprint; and

a second network interface connecting said one or more virtual IDS units to a protected network;

wherein at least one said virtual IDS units comprises at least one threat signature specific to an operating system and is configured to accept only packets that do not match any said threat signature.

17. The apparatus of Claim 16, wherein said fingerprinter employs passive fingerprinting algorithms.

18. The apparatus of Claim 16, wherein said fingerprinter employs active fingerprinting algorithms.

19. The apparatus of Claim 16, wherein, when said fingerprinter identifies more than one target OS fingerprint, said director directs each said packet to said virtual IDS units at substantially the same time.

20. An apparatus for intrusion detection system virtualization, comprising:

means for receiving a stream of packets;

means for fingerprinting each packet in a said stream to identify at least one target operating system (OS) type;

means for directing each said packet to a virtual IDS process corresponding to each said identified target OS type;

means for comparing each said packet to a threat signature set corresponding to each said identified target OS type in said virtual IDS process; and

means for accepting each said packet based on said comparing.

21. The apparatus of Claim 20, wherein said means for fingerprinting comprise means for active fingerprinting.

22. The apparatus of Claim 20, wherein said means for fingerprinting comprise means for passive fingerprinting.

23. The apparatus of Claim 20, wherein said target OS type is the packet source host OS.

24. The apparatus of Claim 20, wherein said target OS type is the packet destination host OS.

25. The apparatus of Claim 20, wherein said target OS type is both the packet source host OS and the packet destination host OS.

26. The apparatus of Claim 20, wherein, when said means for fingerprinting identifies more than one target OS type, said means for directing operates at substantially the same time.

27. A computer system for use in intrusion detection system virtualization, comprising computer instructions for:

receiving a stream of packets;

fingerprinting each packet in a said stream to identify at least one target operating system (OS) type;

directing each said packet to a virtual IDS process corresponding to each said identified target OS type;

comparing each said packet to a threat signature set corresponding to each said identified target OS type in said virtual IDS process; and

accepting each said packet based on said comparing.

28. The computer system of Claim 27, wherein said computer instructions for fingerprinting comprise computer instructions for active fingerprinting.

29. The computer system of Claim 27, wherein said computer instructions for fingerprinting comprise computer instructions for passive fingerprinting.

30. A computer-readable medium storing a computer program executable by a plurality of server computers, the computer program comprising computer instructions for:

receiving a stream of packets;

fingerprinting each packet in a said stream to identify at least one target operating system (OS) type;

directing each said packet to a virtual IDS process corresponding to each said identified target OS type;

comparing each said packet to a threat signature set corresponding to each said identified target OS type in said virtual IDS process; and

accepting each said packet based on said comparing.

31. The computer-readable medium of Claim 30, wherein said computer instructions for fingerprinting comprise computer instructions for active fingerprinting.

32. The computer-readable medium of Claim 30, wherein said computer instructions for fingerprinting comprise computer instructions for passive fingerprinting.

33. A computer data signal embodied in a carrier wave, comprising computer instructions for:

receiving a stream of packets;

fingerprinting each packet in a said stream to identify at least one target operating system (OS) type;

directing each said packet to a virtual IDS process corresponding to each said identified target OS type;

comparing each said packet to a threat signature set corresponding to each said identified target OS type in said virtual IDS process; and

accepting each said packet based on said comparing.

34. The computer data signal of Claim 33, wherein said computer instructions for fingerprinting comprise computer instructions for active fingerprinting.

35. The computer data signal of Claim 33, wherein said computer instructions for fingerprinting comprise computer instructions for passive fingerprinting.

Fig. 1

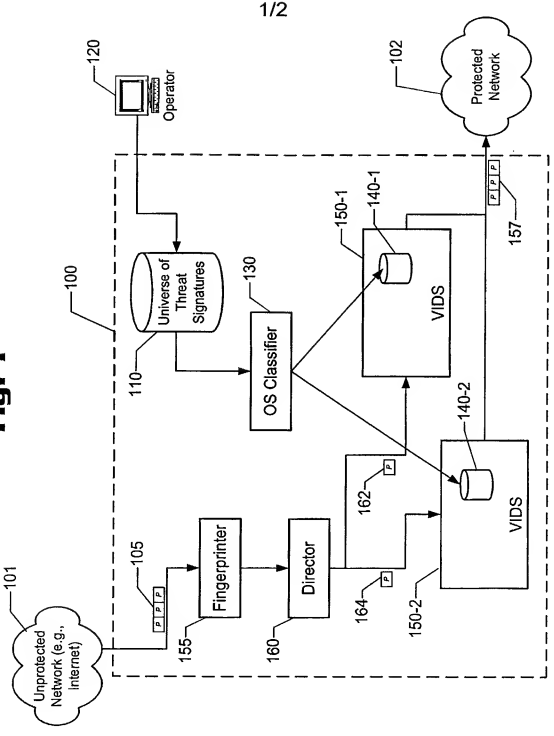
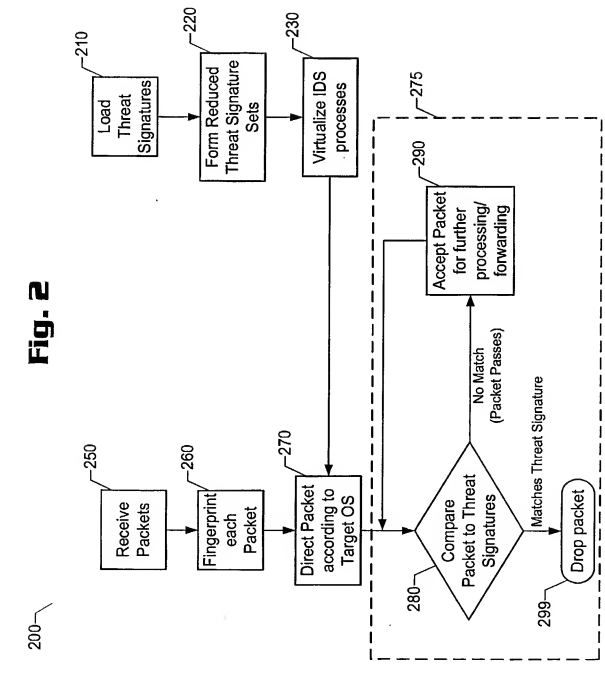

101
Unprotected Network (e.g., Internet)
105
155
Fingerprinter
160
Director
162
164
150-2
VIDS
140-2
150-1
VIDS
140-1
OS Classifier
130
110
Universe of Threat Signatures
100
120
Operator
157
102
Protected Network

Fig. 2


200
210
Load Threat Signatures
220
Form Reduced Threat Signature Sets
230
Virtualize IDS processes
250
Receive Packets
260
Fingerprint each Packet
270
Direct Packet according to Target OS
275
280
Compare Packet to Threat Signatures
No Match (Packet Passes)
290
Accept Packet for further processing/ forwarding
Matches Threat Signature
299
Drop packet

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US2005/011702

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, IBM-TDB, INSPEC, WPI Data, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2003/212910 A1 (ROWLAND CRAIG H ET AL) 13 November 2003 (2003-11-13) figure 3 paragraph [0006] paragraph [0024] paragraph [0031] - paragraph [0032] -----	1-35
E,X	US 2005/086522 A1 (ROWLAND CRAIG H) 21 April 2005 (2005-04-21) figure 3 paragraph [0004] paragraph [0028] - paragraph [0029] -----	1-35
	-/--	

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
6 July 2005	09. 09. 2005

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Raposo Pires, J

Form PCT/ISA/210 (second sheet) (January 2004)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US2005/011702

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	DEBAR H ET AL: "Towards a taxonomy of intrusion-detection systems" COMPUTER NETWORKS, ELSEVIER SCIENCE PUBLISHERS B.V., AMSTERDAM, NL, vol. 31, no. 8, 23 April 1999 (1999-04-23), pages 805-822, XP004304519 ISSN: 1389-1286 page 808, paragraph 3.1.1. page 809, paragraph 3.1.1.2 page 817, paragraph 4 -----	1-35
A	US 2003/188189 A1 (DESAI ANISH P ET AL) 2 October 2003 (2003-10-02) paragraph [0035] paragraph [0038] paragraph [0048] -----	1-35

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US2005/011702

Patent document cited in search report	Publication date	Patent family member(s)		Publication date
US 2003212910 A1	13-11-2003	US	2003196123 A1	16-10-2003
		AU	2003220582 A1	13-10-2003
		CA	2479504 A1	09-10-2003
		EP	1491019 A1	29-12-2004
		WO	03084181 A1	09-10-2003
US 2005086522 A1	21-04-2005	WO	2005041141 A2	06-05-2005
US 2003188189 A1	02-10-2003	NONE		

DERWENT-ACC-NO: 2005-811408

DERWENT-WEEK: 200757

TITLE: Virtualization method for network intrusion detection system for monitoring network traffic, involves comparing each packet to threat signature set corresponding to target operating system type identified by active fingerprinting

INVENTOR: ITHAL R; ITHAL R G

PATENT-ASSIGNEE: CISCO TECH IND[CISCN] , CISCO TECHNOLOGY INC[CISCN]

PRIORITY-DATA: 2004US-832588 (April 27, 2004)

PATENT-FAMILY:

PUB-NO	PUB-DATE	LANGUAGE
WO 2005109824 A1	November 17, 2005	EN
EP 1741265 A1	January 10, 2007	EN
CN 1943210 A	April 4, 2007	ZH

DESIGNATED-STATES: AE AG AL AM AT AU AZ BA BB BG BR BW BY BZ CA CH CN CO CR CU CZ DE DK DM DZ EC EE EG ES FI GB GD GE GH GM HR HU ID IL IN IS JP KE KG KM KP KR KZ LC LK LR LS LT LU LV MA MD MG MK MN MW MX MZ NA NI NO NZ OM PG PH P L PT RO RU SC SD SE SG SK SL SM SY TJ TM TN TR TT TZ UA UG US UZ VC VN YU ZA ZM ZW AT BE BG BW CH CY CZ DE DK EA EE ES FI FR GB GH GM GR HU IE IS IT KE LS LT LU MC MW MZ NA NL OA PL PT RO SD SE SI SK SL SZ TR TZ UG ZM ZW AT BE BG CH CY CZ DE DK EE ES FI F R GB GR HU IE IS IT LI LT LU MC NL PL PT RO SE SI SK TR

APPLICATION-DATA:

PUB-NO	APPL-DESCRIPTOR	APPL-NO	APPL-DATE
WO2005109824A1	N/A	2005WO-US011702	April 5, 2005
CN 1943210A	N/A	2005CN-80011926	April 5, 2005
EP 1741265A1	N/A	2005EP-736313	April 5, 2005
EP 1741265A1	Based on	2005WO-US011702	April 5, 2005

INT-CL-CURRENT:

TYPE	IPC DATE
CIPP	H04L29/06 20060101
CIPP	H04L29/06 20060101

CIPS H04L29/06 20060101

ABSTRACTED-PUB-NO: WO 2005109824 A1

BASIC-ABSTRACT:

NOVELTY - The method involves performing active fingerprinting of each packet in a received stream (105) to identify a target operating system (OS) type. Each packet is directed to a virtual intrusion detection system (IDS) corresponding to each identified target OS type. Each packet is compared to threat signature sets (140-1,140-2) corresponding to each identified OS type and packet is accepted accordingly.

DESCRIPTION - INDEPENDENT CLAIMS are also included for the following:

(1) virtualization apparatus;

(2) computer system for use in network intrusion detection system virtualization;

(3) computer readable medium storing network intrusion detection system virtualization program; and

(4) computer data signal for network intrusion detection system virtualization.

USE - For virtualizing network intrusion detection system (IDS) for monitoring network traffic and for blocking attempted attacks on or intrusion into protected network space e.g. for local area network (LAN), metropolitan area network (MAN), etc.

ADVANTAGE - Enhances the processing efficiency and speed by reducing the set of threat signatures that need to be searched for each packet.

DESCRIPTION OF DRAWING(S) - The figure shows a high level block diagram of the virtualized intrusion detection system (VIDS).

virtualized intrusion detection system (100)

stream of packets (105)

threat signature sets (140-1,140-2)

VIDS (150-1,150-2)

finger printer (155)

CHOSEN-DRAWING: Dwg.1/2

TITLE-TERMS: METHOD NETWORK INTRUDE DETECT SYSTEM MONITOR TRAFFIC COMPARE PACKET THREAT SIGNATURE SET CORRESPOND TARGET OPERATE TYPE IDENTIFY ACTIVE FINGERPRINT

DERWENT-CLASS: T01 W01

EPI-CODES: T01-N02B2B; T01-N02B2C; T01-S03; W01-A06A3;

SECONDARY-ACC-NO:

Non-CPI Secondary Accession Numbers: 2005-672792